Supplementary Information for

Quality score compression improves genotyping accuracy

Y. William Yu[1,2], Deniz Yorukoglu[2], Jian Peng[1,2], Bonnie Berger*[,1,2]

[1]Department of Mathematics, Massachusetts Institute of Technology, Cambridge, MA

[2]Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA

*To whom correspondence should be addressed;
Email: bab@mit.edu

October 23, 2014

A Methods

A.1 Dictionary generation

To generate the dictionary, we compiled a list of commonly occurring 32-mers in a population-size corpus of sequencing reads. For the corpus, we used 194 FASTQ files taken from the 1000 Genomes Project [1], corresponding to paired-end reads from 97 human individuals (see Supplementary Information: Datasets). Read lengths in the dataset ranged from 91-103bp with a total depth of coverage of ∼700x.

To construct the dictionary of k-mers memory efficiently, we modified the Misra-Gries approximate counting algorithm [2]:

Modified Misra-Gries:

1. Initialize a primary hash map A.
   Initialize 65535 hash sets $C_1, \ldots, C_{65535}$.
   Initialize the counter $D = 0$.
2. **update(i): if** $i \in A$, remove i from $C_{A[i]}$, increment $A[i]$ (unless $A[i] = 65535$), and insert i to the new $C_{A[i]}$.
   else if $|A| < m$, insert $(i, 1)$ into A and insert i into C_1.
   else Remove an arbitrary element j from C_D (while $|C_D| = 0$, increment D until nonempty). Remove j from A. Insert $(i, D + 1)$ into A and i into C_{D+1}.
3. **query(i): if** $i \in A$ **then** output $A[i] - D$.
   else output 0

Of note, in contrast to standard Misra-Gries, we use a collection of hash buckets instead of doubly linked lists to minimize cache misses. As we do not care about extremely high counts, we further arbitrarily limit the number of hash buckets to 65535. Memory usage is controlled by maintaining the invariant that the total number of k-mers stored never exceeds the memory-usage parameter m. As with classic Misra-Gries, the approximation error is bounded by the decrement counter D.

After partitioning the space of 32-mers into 256 subsets by the identity of the middle 4 bases, we ran the modified Misra-Gries in multiple passes on each of the subsets. By controlling the approximation ratio of the Misra-Gries algorithm, we guaranteed that all 32-mers in the dictionary occurred at least 200 times in the corpus and that all 32-mers in the corpus that occurred at least 240 times were included in the dictionary. In total, the dictionary contained 2,497,777,248 unique 32-mers. It is available with **Software**.

A.2 Quality score compression

As input, our quality score compression algorithm Quartz requires the dictionary described above, consisting of commonly-occurring k-mers extracted from a population-sized read dataset. The

dictionary is designed in such a way that any given read dataset can be mostly covered from these k-mers within a small Hamming distance, or number of mismatches. Using this dictionary, Quartz compresses the quality scores in any given read dataset by identifying k-mers from each read within a small Hamming distance from other k-mers in the dictionary. Any quality score value corresponding to a position that is concordant with at least one supporting k-mer is set to a default value, whereas the quality score value at any position that is divergent from all supporting k-mers is kept. This coarse representation greatly reduces the storage requirement for the quality scores of read datasets since the smoothed quality score values are substantially more compressible than the original values.

Our implementation was written in C++11 and compiled with GCC 4.7.2 with all relevant optimizations enabled. Because each read can be independently processed, Quartz is trivially parallelized using OpenMP, achieving speedups nearly linear to the number of cores used, up to the disk I/O bound.

A.3 Parameter selection

Efficiency concerns sufficiently dictate the parameters of k-mer length and Hamming distance that we have hard-coded optimal choices into the software. For the experiments presented in the paper, we selected the k-mer length parameter, k, to be 32bp. There are three main criteria we considered when selecting the k-mer length within the Quartz quality score compression framework:

1. k-mers should be long enough to ensure that the number of all possible k-mers is much larger than the number of unique k-mers in the genome, so as to ensure incidental collisions between unrelated k-mers are rare.

2. Since within the Quartz framework, k-mer neighbors are defined to be within a Hamming distance of 1, k-mers should also be short enough to allow the probability that a k-mer contains more than one sequencing error to be low; this criterion is to ensure that k-mer sequences within reads originating from the same genomic region are highly likely to be detected as neighbors.

3. k-mer length should ideally be a multiple of four, since a 4bp length DNA sequence can be represented by a single byte.

A 32bp long k-mer satisfies all three of these criteria; it is represented by a single 64-bit integer, with a relatively low probability of containing more than one sequencing error with Illumina sequences, as well as resulting in few k-mer collisions. Experimental results demonstrating that other parameter values are inefficient can be found in Table S.4. A more rigorous analysis scheme by several of the authors of this paper is also available [3].

For the experimental results presented, the default replacement value (Q) for smoothed quality scores was selected as 50. In the datasets we studied, the average quality score excluding the special

value of 2, which we did not modify, was at least 35. Recall that Quartz only replaces a quality score within a 32-mer if all Hamming neighbors agree on a position. Thus, assuming that all 32-mers within a read are variants of 32-mers in the dictionary, the method only incorrectly smooths a quality score if there are two read errors: one for the quality score and one at one of the other 31 locations. Then, as a first order approximation, the error probability for a smoothed location is $31 \times 10^{-3.5} \times 10^{-3.5} < 10^{-5}$, justifying our choice of $Q = 50$. However, this is a trivially adjusted parameter, and we have explored several replacement values in Fig. S.9.

A.4 Hamming neighbor search

As Hamming neighbors are defined as all k-mers in the dictionary that differ from a query k-mer by no more than 1 base substitution, we need to be able to quickly search for all $3k + 1$ potential Hamming neighbors in the dictionary. The nave approaches of using a sorted list or hash tables suffer from the shortcomings of being respectively either CPU or memory inefficient. Additionally, both naïve approaches incur many cache misses.

We instead used an approach inspired by locality sensitive hashing. Recall that an (R, cR, P_1, P_2)-sensitive LSH family F of hash functions $h : M \to S$ is defined if $\forall p, q \in M$, a uniformly random $h \in F$ satisfies two conditions:

1. if $||p - q|| \leq R$, then $Prob(h(p) = h(q)) \geq P_1$, and
2. if $||p - q|| \geq cR$, then $Prob(h(p) = h(q)) \leq P_2$.

Note that projecting k-mers onto random $\frac{k}{2}$-length subsequences forms a $(1, c, \frac{1}{2}, 2^{c-1})$-sensitive LSH hash family under the Hamming metric. Further note that any such projection h comes with an orthogonal projection h. Because the Hamming metric is discrete, all Hamming neighbors of a k-mer must match under one of the two projections. Thus, by double hashing, all Hamming neighbors can be found by looking in just two hash buckets, one for each hash function, giving much better cache-efficiency. Additionally, as genomic sequences are not chosen in an adversarial manner, instead of using random projections, projecting onto the front and back halves of the k-mer suffices in practice.

A.5 Memory requirements

As the dictionary is static and read-only, both hash tables can be efficiently implemented by sorting lexicographically first by the key (i.e. the projected half) and then by the rest, saving pointers to the first dictionary entry in each "bucket", and then discarding the projected half of each k-mer—as all k-mers are only accessed through the bucket pointers, the hash key is implied. This scheme also allows for binary search within each bucket, further speeding up lookups. For the default $k = 32$, we use an array of length 2^{32} of unsigned 32-bit integers for the pointers and an array of unsigned

32-bit integers of length $|D|$, the dictionary size, for the remaining half of the k-mers not specified by the pointers. Thus, total memory requirement for one such hash table is $(2^{32} + |D|) \cdot 4$ bytes, and since two hash tables are needed, $(2^{32} + |D|) \cdot 8$ bytes. For $|D| = 2,497,777,248$ as for the human genome, Quartz requires ∼54 GiB of memory.

A.6 Experimental design

The existence of high quality trio-validated SNP calls makes NA12878 an ideal candidate for testing genotyping accuracy. We used the SRR622461_1.filt.fastq, SRR622461_2.filt.fastq, and SRR622461.filt.fastq data files from the 1000 Genomes Project [1] for the unmapped, raw reads and the variant list CEUTrio.HiSeq.WGS.b37.bestPractices.phased.hg19.vcf from the Broad GATK bundle as a gold standard for variant locations [4].

Quartz was used to generate smoothed FASTQ files with most quality scores reset to Q. To measure differences in entropy, both the original and modified quality scores were compressed with BZIP2, GZIP, 7z (PPMd), XZ (LZMA2), and FASTQZ [5]. The resulting file sizes were used to compute a rough estimate of the number of bits of storage needed per quality score. We ran BWA aln/sampe/samse (version 0.7.5a-r405) [6] and Bowtie 2 (version 2.1.0) [7] with all default options to map the relevant FASTQ files. We used two state-of-the-art variant callers for estimating genotyping accuracy: GATK UnifiedGenotyper [4] and SAMtools mpileup [8]. We ran the GATK UnifiedGenotyper (version 2.4.7-g5e89f01) with default parameters. However, because Quartz resets base qualities to high, it does not interact well with the Base Alignment Quality filter [9]; thus, we ran SAMtools mpileup (version 0.1.18-dev r982:313) with all default options except for the Base Alignment Quality filter (varFilter option "-2 0").

To measure the relative downstream genotyping accuracy, we computed a rescaled receiver operating characteristic (ROC) curve on variant locations using the ROCR package [10]. Recall that variant callers do not simply give a list of variant calls. Rather, they give a list of variant calls along with variant call confidences. In order to provide a final list, some variant call confidence threshold must be chosen, at which point one can figure out the true and false positive ratio. The ROC curve is precisely the parameterized curve that results from varying that confidence threshold, and allows for comparing two methods without having to arbitrarily choose a single confidence threshold. However, in considering variant callers as binary classifiers, the true negative rate of correctly called non-SNPs easily dwarfs everything else as most of the genome is called as not a SNP. As such, to compare two (or more) sets of variant call locations, we took the union of those call positions as the domain. The ROC curves were then computed against a gold standard classification of domain elements. Note that this rescaling implies that ROC curves are not comparable between different plots.

All timing benchmarks were run on a Debian GNU/Linux 6.0.8 system with dual Intel Xeon X5690 processors and 94 GiB RAM. However, to ensure comparability, we disabled parallelization,

artificially limiting all programs to a single core unless specified otherwise.

B Datasets

B.1 Dictionary generation

The dictionary used in this paper was generated by finding commonly occurring 32-mers in paired-end reads from recent 1000 Genomes Project FASTQ files from 97 individuals [11, 1]. We used two FASTQ files associated with each sample corresponding to the two mate pairs of all paired end reads for which both mate pairs passed filtering/quality control. Those FASTQ files were downloaded from URLs of the form

ftp://ftp.1000genomes.ebi.ac.uk/vol1/ftp/data/[SAMPLE_NAME]/sequence_read/[FASTQ_FILE]_1.filt.fastq.gz
ftp://ftp.1000genomes.ebi.ac.uk/vol1/ftp/data/[SAMPLE_NAME]/sequence_read/[FASTQ_FILE]_2.filt.fastq.gz

where [SAMPLE_NAME] and [FASTQ_FILE] are given below.

SAMPLE_NAME	FASTQ_FILE
NA19649	SRR189825
NA19657	SRR068163
NA19658	SRR189828
NA19663	SRR068160
NA19664	SRR068164
NA19669	SRR189827
NA19670	SRR189826
NA19720	SRR350122
NA19914	SRR350098
NA20362	ERR257982
NA20821	ERR257967
NA21125	ERR055396
NA21137	ERR055395
NA18966	ERR234335
NA18978	ERR234334
NA19351	ERR229817
NA19764	ERR229816
NA20357	ERR229818
NA20359	ERR229819
NA20503	ERR229821
NA20507	ERR229820
NA20513	ERR229822
NA20514	ERR229823
NA18528	ERR234331
NA18531	ERR234333
NA18533	SRR189816
NA18537	ERR034771
NA18572	ERR034774
NA18609	ERR034777
NA18611	ERR034778
NA18991	ERR052929
NA18488	SRR189829
NA18501	ERR234330

SAMPLE_NAME	FASTQ_FILE
NA18870	ERR234332
NA18881	ERR257965
NA19092	SRR189830
NA19093	ERR229810
NA19137	ERR229811
NA19138	ERR229812
NA19141	SRR211276
NA19143	SRR211272
NA19175	ERR229813
NA19210	ERR229815
NA19222	ERR229814
NA20900	ERR257972
NA21104	SRR768143
NA21120	ERR126299
NA21122	ERR125594
NA21123	ERR125595
NA21127	ERR257973
NA18504	SRR350142
NA18912	SRR350153
NA19200	SRR352199
NA19160	SRR352222
NA19171	SRR359061
NA18968	SRR359062
NA19204	SRR359064
NA18975	SRR359070
NA18981	SRR359083
NA18971	SRR359095
NA19131	SRR359096
NA19152	SRR359097
NA19119	SRR359106
NA18976	SRR359110
NA18974	SRR360136

SAMPLE_NAME	FASTQ_FILE
NA06994	SRR385751
NA11930	ERR233226
NA11932	ERR233225
NA11933	ERR233227
NA11992	SRR385759
NA11994	SRR385753
NA12003	SRR385756
NA12046	ERR239333
NA12154	SRR385769
NA12155	SRR385762
NA12234	ERR233302
NA12249	SRR211278
NA12282	ERR233301
NA12283	ERR239334
NA12286	ERR234321
NA12340	SRR075006
NA12716	ERR257986
NA12717	ERR257987
NA12748	ERR234323
NA12750	ERR257985
NA12751	ERR257988
NA12761	ERR257989
NA12827	ERR234322
NA12829	ERR234325
NA12830	ERR234324
NA12842	ERR234327
NA12843	ERR234326
NA12878	SRR622461
NA12889	ERR234328
NA12890	ERR234329
NA18969	SRR211274
NA18970	SRR211273

B.2 NA19240

In addition to the NA12878 datasets we used in the main body of the paper, to ensure that our results are widely applicable, we also performed several analyses on NA19240, using

SRR794330_1.filt.fastq, SRR794330_2.filt.fastq, SRR794330.filt.fastq,
SRR794336_1.filt.fastq, SRR794336_2.filt.fastq, SRR794336.filt.fastq

for the raw unaligned reads. As a gold standard list of variant locations, we used

NA19240_lcl_SRR832874.wgs.COMPLETE_GENOMICS.20130401.snps_indels_svs_meis.high_coverage.genotypes.vcf

from the Complete Genomics variant calls using CGAPipeline 2.2.0.26, generated on a high coverage ($\sim 80x$) dataset using CGA Tools [12].

C Supplemental Results

Quality score recalibration decreases compressibility. Although there are some superficial similarities to quality score recalibration methods, in that both they and Quartz use k-mer profiles and improve downstream variant calling accuracy, we demonstrate here in Table S.1 the major difference: Quartz is a compression algorithm, and decreases space requirements, whereas the GATK [4] recalibration increases storage space required. We demonstrate this on the quality scores for NA12878 after mapping by Bowtie 2 [7]. As a note, because the GATK recalibration is performed post-mapping and sorting by read position, the compression ratios differ slightly from those given in Table S.2 for unmapped reads. Although Quartz is performed pre-mapping, here we give the compression results after mapping to ensure comparability to the GATK recalibration. Finally, total time required for the listed steps is given, demonstrating that Quartz compression is much faster than GATK recalibration.

Table S.1: Time requirements and effect on storage of Quartz compression and/or GATK recalibration on NA12878, measured by CPU-time and BZ2 file size. Of note, recalibration significantly increases storage size. Furthermore, recalibration is much slower than Quartz compression.

	CPU-time	Bits/Q
Original	0	1.4159
GATK recalibrated	22,686	1.9854
Quartz compressed	2,696	0.3601

Timing and compression benchmarks. In Table S.2 we present the compression ratios of off-the-shelf compressors; Quartz smoothing reduces the amount of storage space needed by at least a factor of three. Additionally, also shown in the table is the performance of FASTQZ [5] on quality scores, a state-of-the-art FASTQ compressor that uses a clever lossless encoding of quality scores before compression, over which Quartz also demonstrates improvement.

As stated in the paper, our implementation of Quartz costs a negligible amount of CPU time compared to mapping and variant calling. In Table S.3 we give single-threaded timing benchmarks for Quartz, mapping, and variant calling on NA12878 using the Bowtie 2 [7] + GATK [13, 4, 14] and BWA [6] + SAMtools [8] pipelines presented in Figure 1 of the paper, both with and without Quartz compression. Quartz is orders of magnitude faster than mapping and genotyping.

Table S.2: Compression benchmarks (bits per quality score) on NA12878. After smoothing using Quartz, the number of bits needed to store a quality value was cut by over two thirds for post-processing with lossless compressors, including both off-the-shelf text compressors and FASTQZ, which is designed specifically for genomic data.

	Original	Quartz-smoothed
GZIP	1.7025	0.5043
BZIP2	1.3842	0.3564
7zip (PPMd)	1.2620	0.3676
XZ (LZMA2)	1.4353	0.4010
FASTQZ	1.1434	0.3089

Also in Table S.3 are presented timing benchmarks for off-the-shelf (lossless) compressors on just the quality scores, with and without Quartz compression. Of particular note is the fact that smoothing the quality scores using Quartz nearly doubles the speed of off-the-shelf compressors. NB: because the FASTQZ code does not have an option to only compress quality scores without compressing sequence, we could not separate the two compression times and so have not included FASTQZ in the timing table, as it would be an unfair comparison for FASTQZ, which takes longer than off-the-shelf compressors, but has to compress sequences as well.

Table S.3: Timing benchmarks (in seconds) on NA12878 taken from the Bowtie 2 + GATK and BWA + SAMtools pipelines. Smoothing using Quartz is orders of magnitude faster than both mapping and variant calling steps. Furthermore, Quartz smoothing also significantly increases the speed of off-the-shelf compressors.

		Original Quality Scores	Quartz-smoothed
Quartz		0	2,696
Mapping	Bowtie 2	59,499	59,554
	BWA	116,411	111,023
Variant calling	GATK	23,721	22,261
	SAMtools	24,121	24,190
Off-the-shelf (lossless) compressors	GZIP	817	351
	BZIP2	937	505
	7zip (PPMd)	1,372	565
	XZ (LZMA2)	12,600	6,551

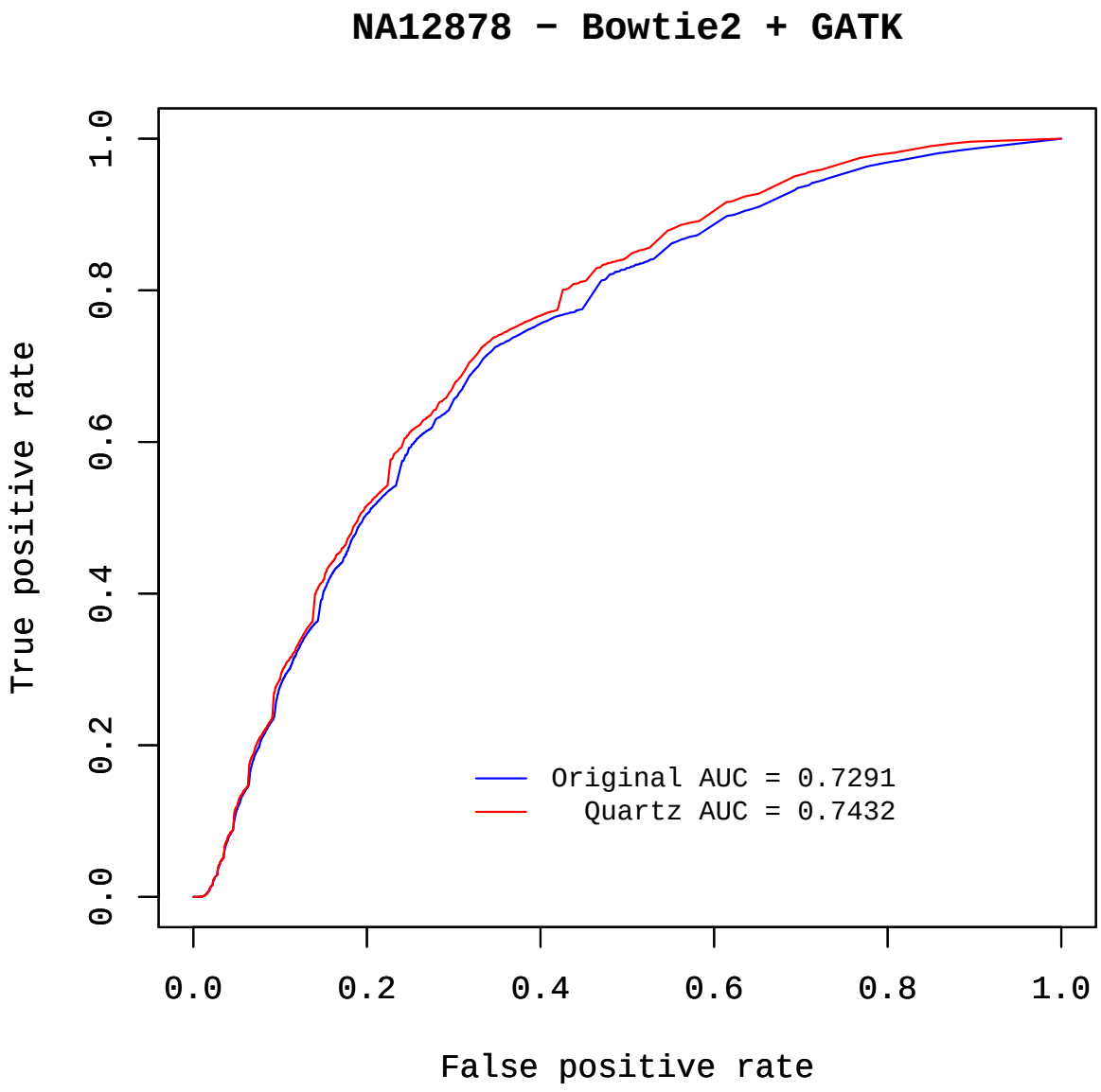
Additionally, there are two hard-coded parameters for Quartz compression, including the choice of k-mer length $k = 32$ and the restriction of Hamming neighbors to one substitution. We benchmark in Table S.4 the effects of adjusting k-mer length and Hamming distance. For k-mer length, we benchmark only $k = 16$, as $k = 64$ requires infeasible memory requirements for most current machines, and for Hamming distance, we explore only distance 2, as CPU time increases exponentially with distance. As neither option performs well, the authors feel justified in providing an efficient implementation with hard-coded k-mer length and Hamming distance parameters.

Table S.4: Timing (in CPU-seconds) and compression (in bits per quality value of the BZ2 file) benchmarks on NA12878 FASTQ files after adjusting k-mer length and Hamming distance. Both adjustments cause significant drops in both compression and speed, and thus are not available as options in the published code.

	Bits/Q	CPU-seconds
Quartz, k-mer length = 32, Hamming distance = 1	0.3564	2,696
Quartz, k-mer length = 16, Hamming distance = 1	1.7514	16,208
Quartz, k-mer length = 32, Hamming distance = 2	1.2844	140,128

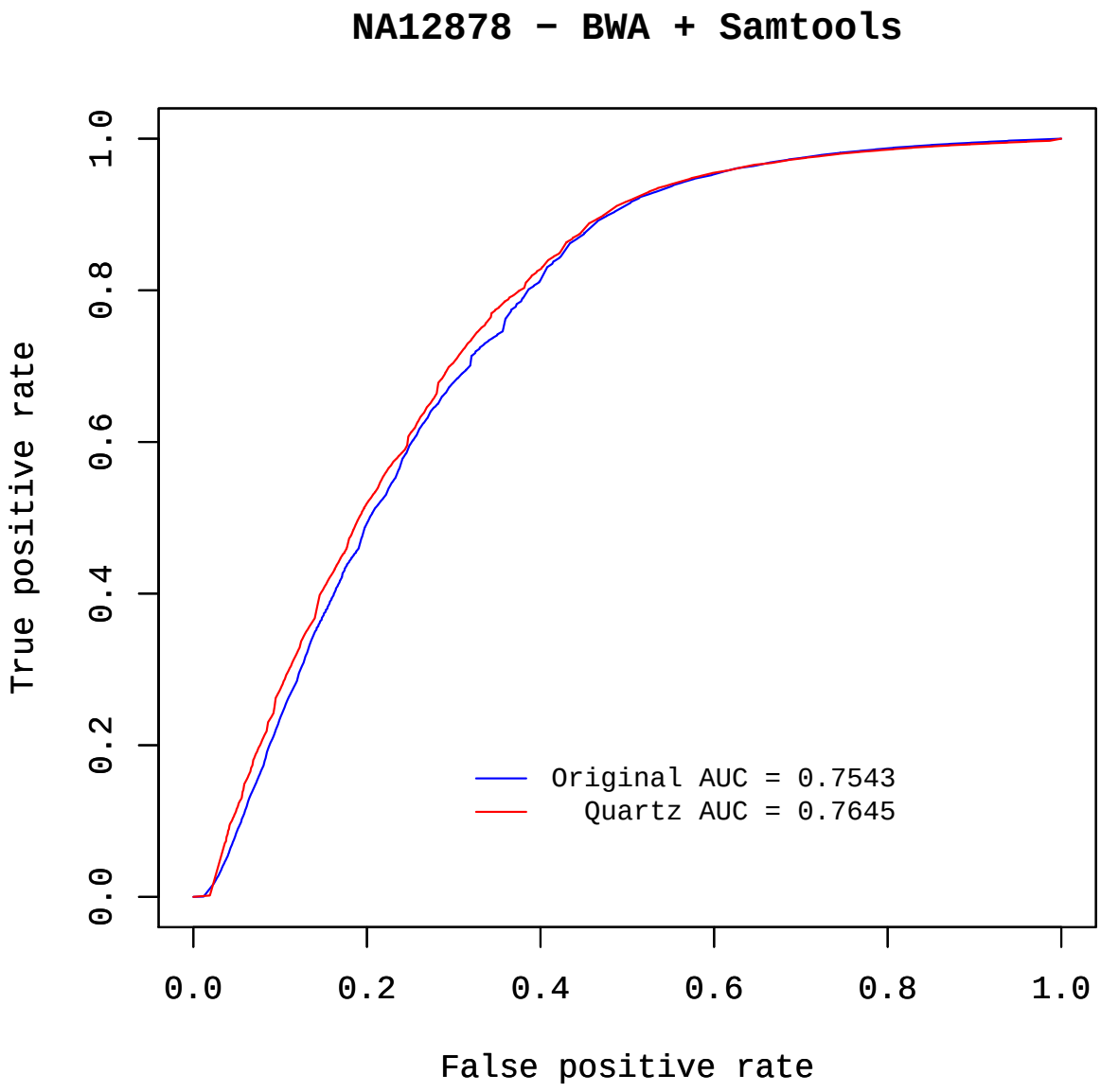
Quartz improves downstream genotyping accuracy. Here we demonstrate that for four different genotyping pipelines that we have tested, Quartz improves to varying degrees the downstream variant-calling accuracy over that of the uncompressed quality scores. Either with or without Quartz quality score compression with a default quality set to 50, NA12878 FASTQ files were aligned with either Bowtie 2 [7] or BWA [6], and variant positions were called with either SAMtools mpileup [8] or the GATK UnifiedGenotyper [13, 4, 14].

Resulting scaled ROC curves with corresponding AUC values are given in Figures S.1, S.2, S.3, and S.4. Note that we have reproduced the results from Figure 2 of this paper, collecting all pipelines together here for easy comparison.



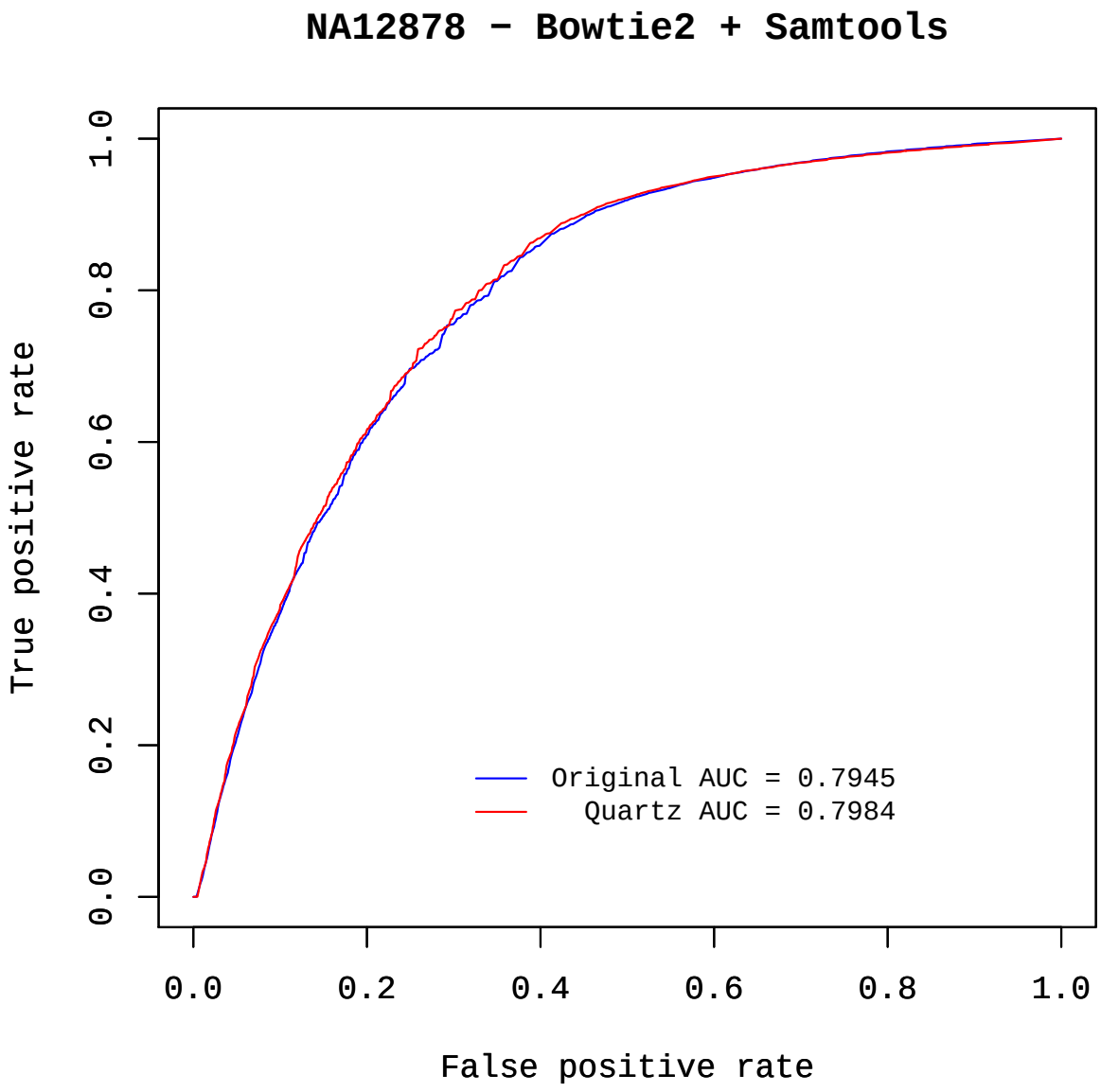
	Original	Quartz
False positives in million highest quality variant calls	24,503	23,383

Figure S.1: Downstream variant calling before and after Quartz compression on the Bowtie 2 + GATK UnifiedGenotyper pipeline for NA12878. Additionally, we give the number of false positives in the million highest quality variant calls.



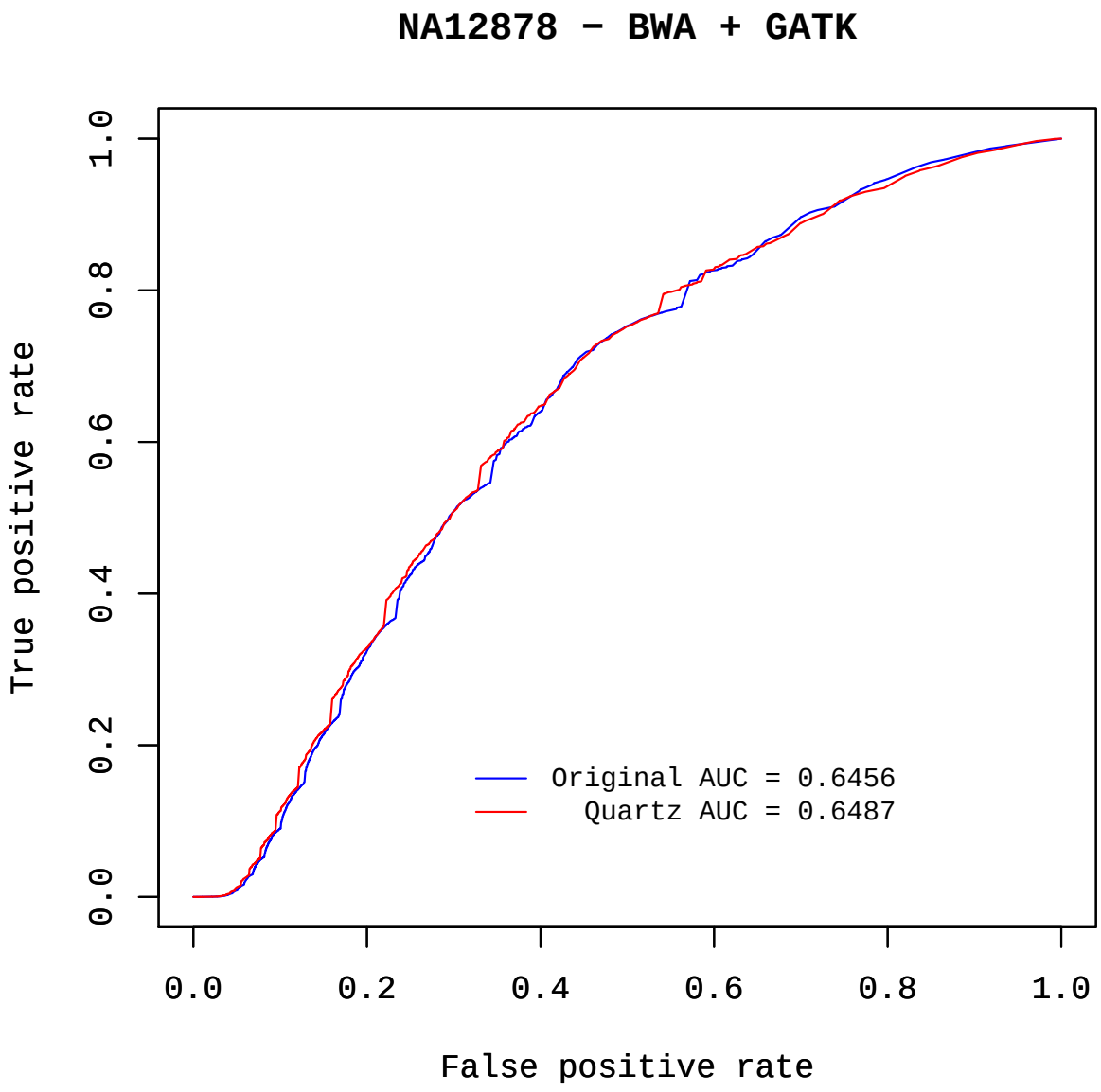
	Original	Quartz
False positives in million highest quality variant calls	33,539	30,042

Figure S.2: Downstream variant calling before and after Quartz compression on the BWA + SAMtools pipeline for NA12878. Additionally, we give the number of false positives in the million highest quality variant calls.



	Original	Quartz
False positives in million highest quality variant calls	16,739	16,077

Figure S.3: Downstream variant calling before and after Quartz compression on the Bowtie 2 + SAMtools pipeline for NA12878. Additionally, we give the number of false positives in the million highest quality variant calls.



	Original	Quartz
False positives in million highest quality variant calls	48,264	48,080

Figure S.4: Downstream variant calling before and after Quartz compression on the BWA + GATK UnifiedGenotyper pipeline for NA12878. Additionally, we give the number of alse positives in the million highest quality variant calls.

Discarding all quality scores decreases variant-calling accuracy. We additionally ran a control where we replaced *all* quality scores with 50, effectively discarding all of them. This causes a severe drop in genotyping accuracy using the Bowtie 2 + GATK pipeline (Figure S.5), demonstrating that retaining some of the quality scores is necessary.

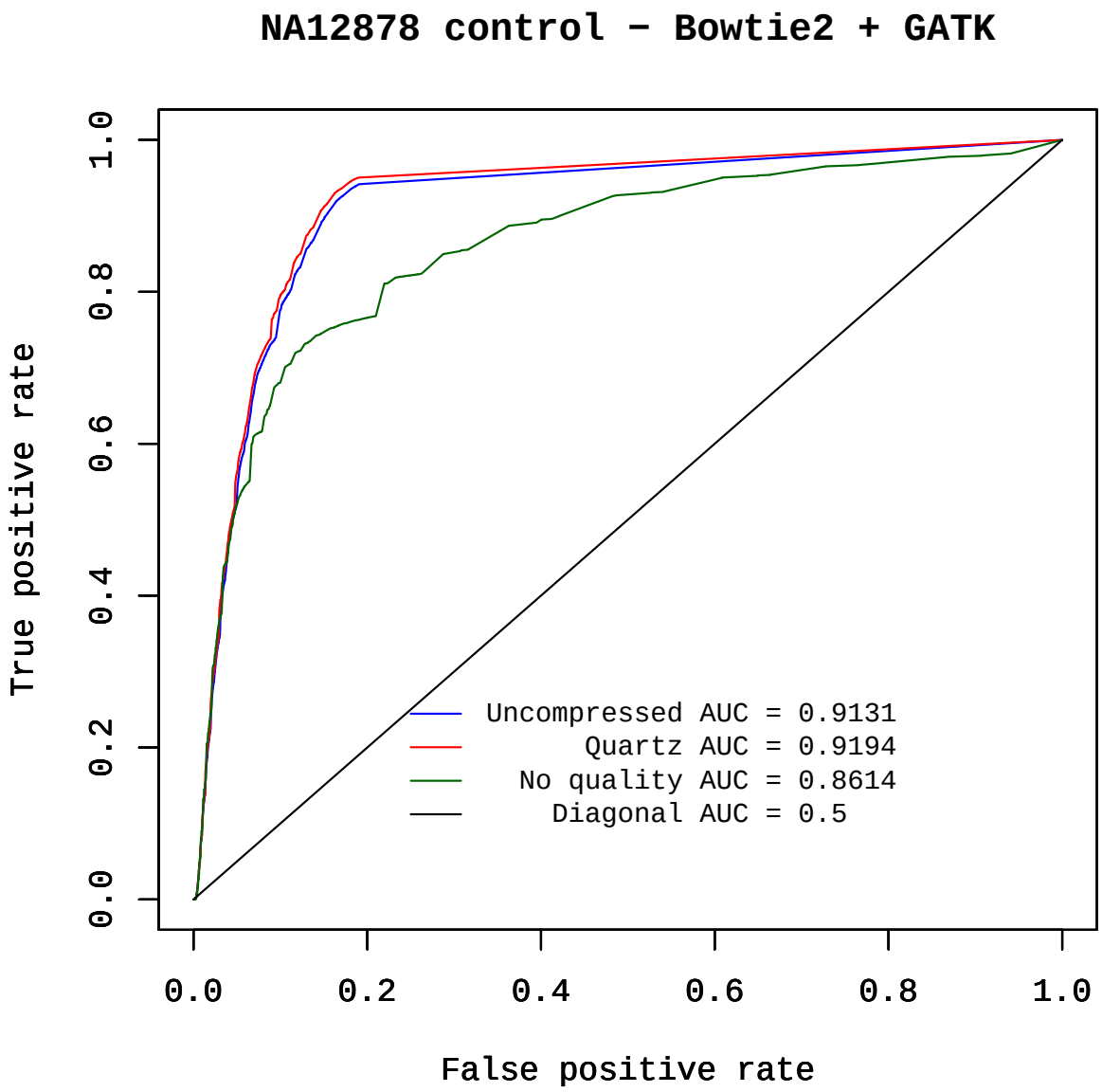
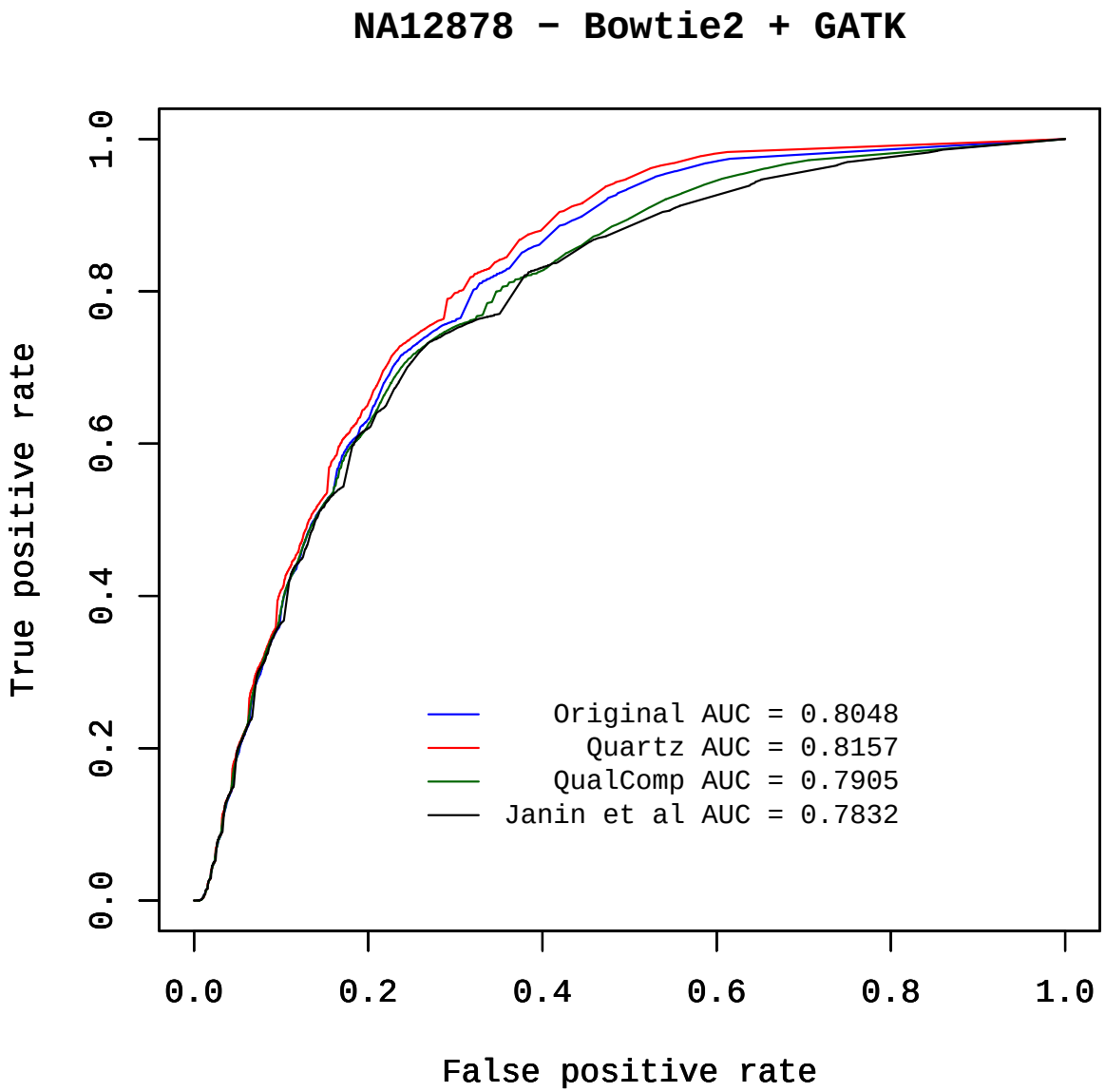


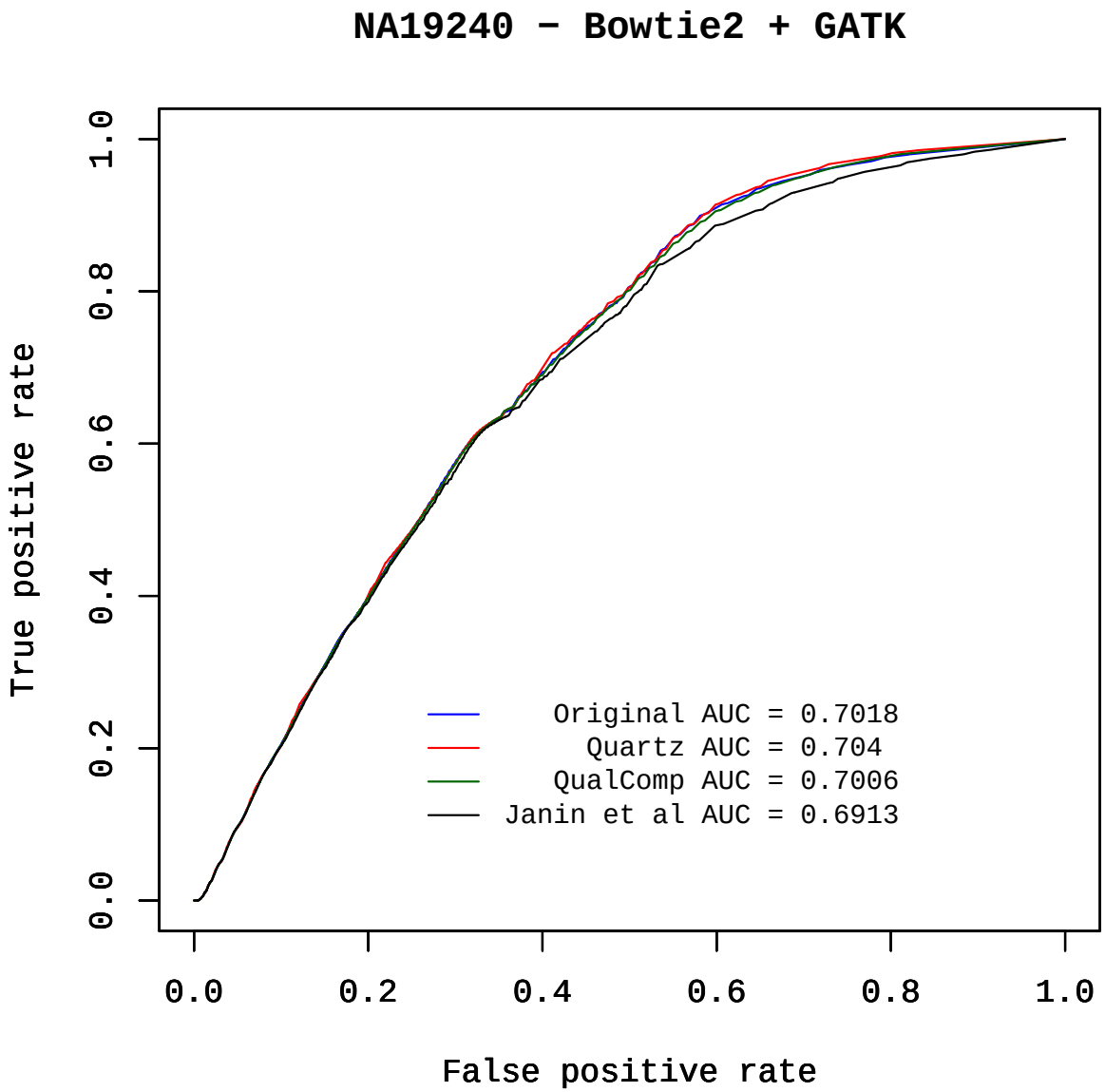
Figure S.5: Control. Discarding all quality scores results in much lower genotyping accuracy. Note that the impact of ROC curve rescaling is most apparent in this figure because of the large numbers of false positives called when all quality scores are discarded.

Quartz out-performs existing lossy compression methods in terms of compression ratio, speed, and genotyping accuracy. Here we show that while maintaining superior compression levels and improving variant-calling accuracy, Quartz offers an order of magnitude improvement in speed compared to other state-of-the-art lossy compression methods in the literature. Quality scores in the FASTQ files were first compressed using one of Quartz, QualComp [15], or Janin et al [16], aligned with Bowtie 2 [7], and then GATK UnifiedGenotyper [13, 4, 14] was used for genotyping. Resulting scaled ROC curves, total single-threaded processing time for compression/decompression, and resulting size of quality scores are given in Figures S.6 and S.7. As Quartz and Janin et al. only alter quality scores in-place to improve compressibility, they were compressed using Bzip2, whereas QualComp has its own compressed file format. Quartz was run with default quality 50. QualComp was run with bits per read set to 60. The method of Janin et al. was run with parameters c=0, s=2, r=39.



Method	Bits/Q	Time (s)	AUC
Uncompressed	8	N/A	0.8048
Quartz	0.3564	2,696	0.8157
QualComp	0.5940	33,316	0.8053
Janin et al.	0.5376	164,702	0.8019

Figure S.6: Quartz compared to QualComp and the method of Janin et al. on NA12878. Quartz is able to simultaneously improve compression ratio, speed, and genotyping accuracy.



Method	Bits/Q	Time (s)	AUC
Uncompressed	8	N/A	0.7018
Quartz	0.5300	2,361	0.7040
QualComp	0.6000	22,898	0.7006
Janin et al.	1.5743	136,207	0.6913

Figure S.7: Quartz compared to QualComp and the method of Janin et al. on NA19240. Quartz is again able to simultaneously improve compression ratio, speed, and genotyping accuracy.

The improvement in accuracy from Quartz is consistent across chromosomes. Although the improvement in accuracy over the uncompressed is easily visible for NA12878, it is less so for NA19240, likely because the gold standard variant calls being used are not as well characterized. However, by demonstrating consistent improvements in variant calling accuracy for all chromosomes, our assertion of increased accuracy is substantiated in Tables S.5 and S.6.

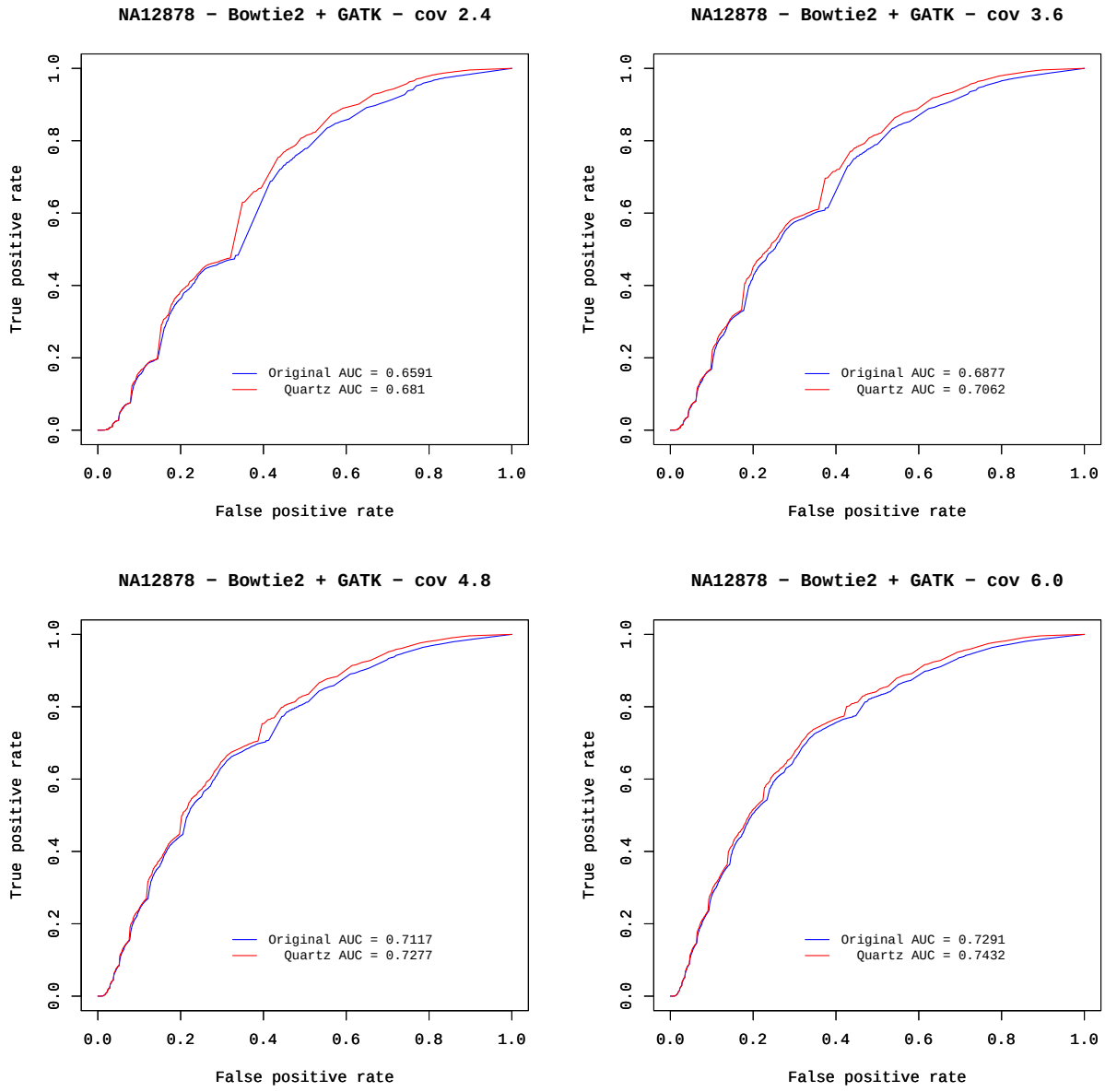
Table S.5: NA12878 ROC AUC values for uncompressed vs Quartz, broken down by chromsome, after using the Bowtie 2 + GATK Unified Genotyper pipeline.

Chromosome	Uncompressed AUC	Quartz AUC	Quartz Improvement
1	0.7893	0.8105	0.0212
2	0.7912	0.8140	0.0228
3	0.7878	0.8110	0.0232
4	0.7940	0.8171	0.0231
5	0.7922	0.8125	0.0203
6	0.7798	0.8036	0.0238
7	0.7829	0.8063	0.0235
8	0.7818	0.8017	0.0199
9	0.7760	0.7980	0.0220
10	0.7778	0.7993	0.0215
11	0.7720	0.7954	0.0234
12	0.7799	0.8006	0.0207
13	0.7923	0.8151	0.0228
14	0.7855	0.8059	0.0204
15	0.7841	0.8075	0.0234
16	0.7601	0.7889	0.0282
17	0.7738	0.7986	0.0248
18	0.7823	0.8064	0.0242
19	0.7407	0.7687	0.0280
20	0.7687	0.7930	0.0243
21	0.7367	0.7696	0.0329
22	0.7413	0.7703	0.0290
X	0.8118	0.8293	0.0176
Mean improvement (w/ standard error): 0.0235 ± 0.0007			

Table S.6: NA19240 ROC AUC values for uncompressed vs Quartz, broken down by chromsome, after using the Bowtie 2 + GATK Unified Genotyper pipeline.

Chromosome	Uncompressed AUC	Quartz AUC	Quartz Improvement
1	0.7255	0.7285	0.0030
2	0.7286	0.7323	0.0037
3	0.7227	0.7275	0.0048
4	0.7012	0.7062	0.0050
5	0.7209	0.7239	0.0030
6	0.7183	0.7221	0.0038
7	0.7112	0.7153	0.0041
8	0.7157	0.7208	0.0051
9	0.7302	0.7356	0.0054
10	0.6959	0.7018	0.0059
11	0.7148	0.7186	0.0038
12	0.7270	0.7311	0.0041
13	0.7088	0.7154	0.0066
14	0.7245	0.7294	0.0049
15	0.7362	0.7385	0.0023
16	0.7078	0.7082	0.0004
17	0.7216	0.7290	0.0074
18	0.7244	0.7306	0.0062
19	0.6965	0.7004	0.0039
20	0.7354	0.7391	0.0037
21	0.6588	0.6644	0.0056
22	0.7174	0.7218	0.0044
X	0.7470	0.7505	0.0035
Mean improvement (w/ standard error): 0.0044 ± 0.0003			

Quartz shows greater improvement in genotyping accuracy at lower coverage read datasets. The reads for NA12878 shown in all figures within this paper except for Figure S.8 had a total depth-of-coverage of about 6. For Figure S.8, the reads were subsampled, preserving mate pairs, and the effect on downstream variant calling was assessed. Although the ROC curves and absolute AUCs are not comparable because of rescaling, we can compare relative differences in AUC.



Coverage	Percent AUC increase
2.4	3.323
3.6	2.690
4.8	2.248
6.0	1.934

Figure S.8: ROC curves for subsampled lower-coverage versions of the NA12878 dataset. Relative improvements in AUC given by Quartz using the Bowtie 2 + GATK UnifiedGenotyper pipeline are given in the table.

Varying the default replacement quality value can slightly affect the downstream variant calling accuracy. While all the results in this paper using a replacement quality value $Q = 50$, this parameter is trivially adjusted and for appropriately high values, has no effect on either compression or CPU-time. Indeed, as suggested in the Online Methods, the optimal choice for this parameter can be related to the average quality value of the dataset at hand. In Figure S.9, we explore the result on downstream variant calling accuracy from varying that value. While the choice of $Q = 40$ was slightly worse, $Q = 45$ and $Q = 50$ were nearly indistinguishable. A more careful examination of the slight differences reveals that $Q = 50$ results in slightly more false positives, accounting for the marginal difference between the latter two in AUC.

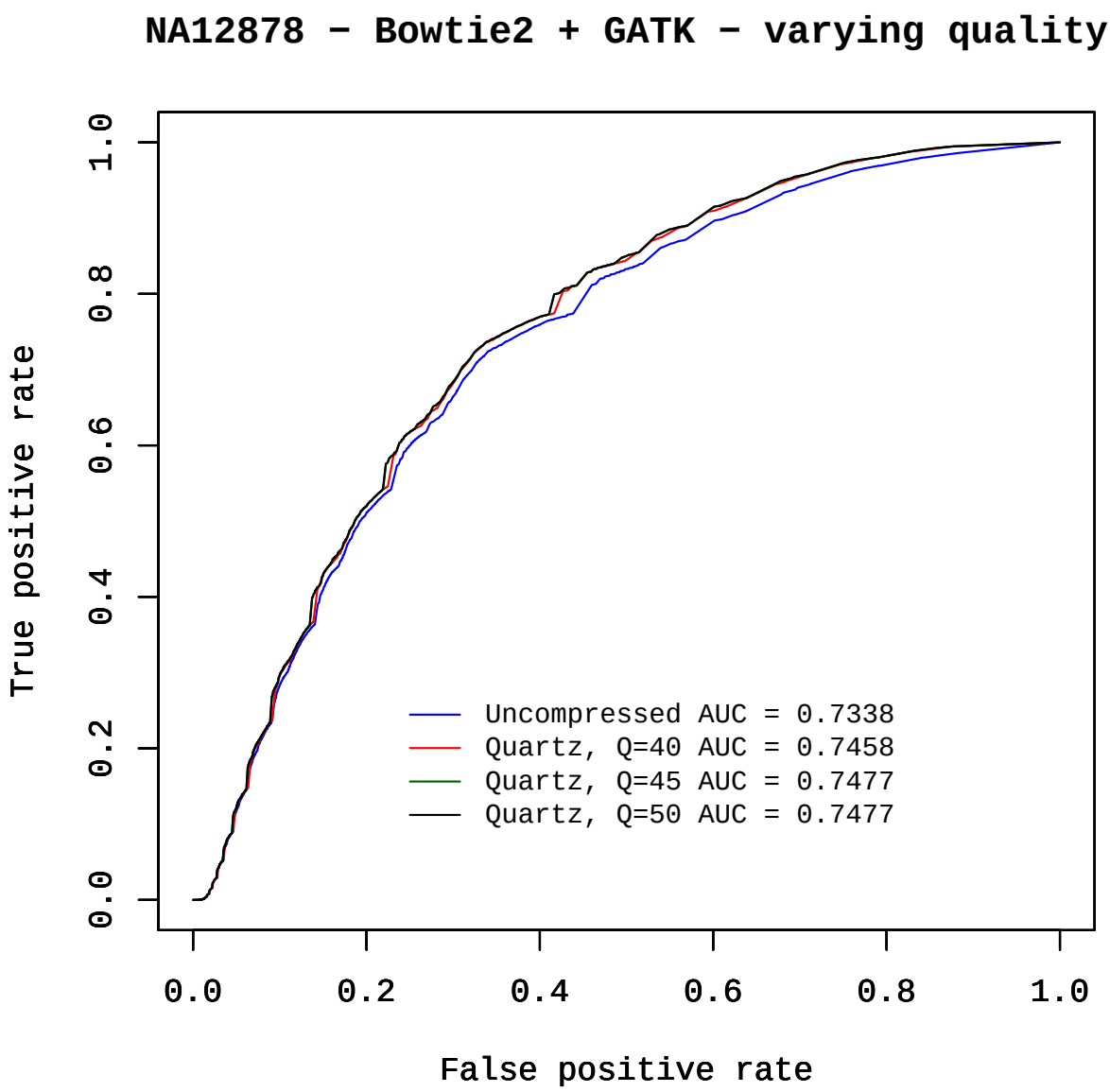


Figure S.9: Downstream variant calling before and after Quartz compression on the Bowtie 2 + GATK UnifiedGenotyper pipeline for NA12878, with varying replacement quality values.

Quartz preserves indel calling accuracy. All of the other variant calling results in this paper have been dominated by SNP-calling accuracy—GATK UnifiedGenotyper only gives SNP calls and although Samtools mpileup gives other variant types, the calls made are predominantly SNPs. Because Quartz is designed with Hamming distance 1 in mind, substitutions are indeed the most appropriate variants for Quartz to handle. However, as Quartz ignores (and hence preserves) quality scores for k-mers greater than distance 1 from the dictionary, Quartz should not at all affect the accuracy of calling indels. In Figures S.11 and S.10 we show the preliminary results with and without Quartz compression for indel calls made using Samtools mpileup. As expected, Quartz does not decrease indel calling accuracy, and even appears to slightly improve the AUCs for these two experiments, so Quartz at least preserves indel accuracy while achieving high compression.

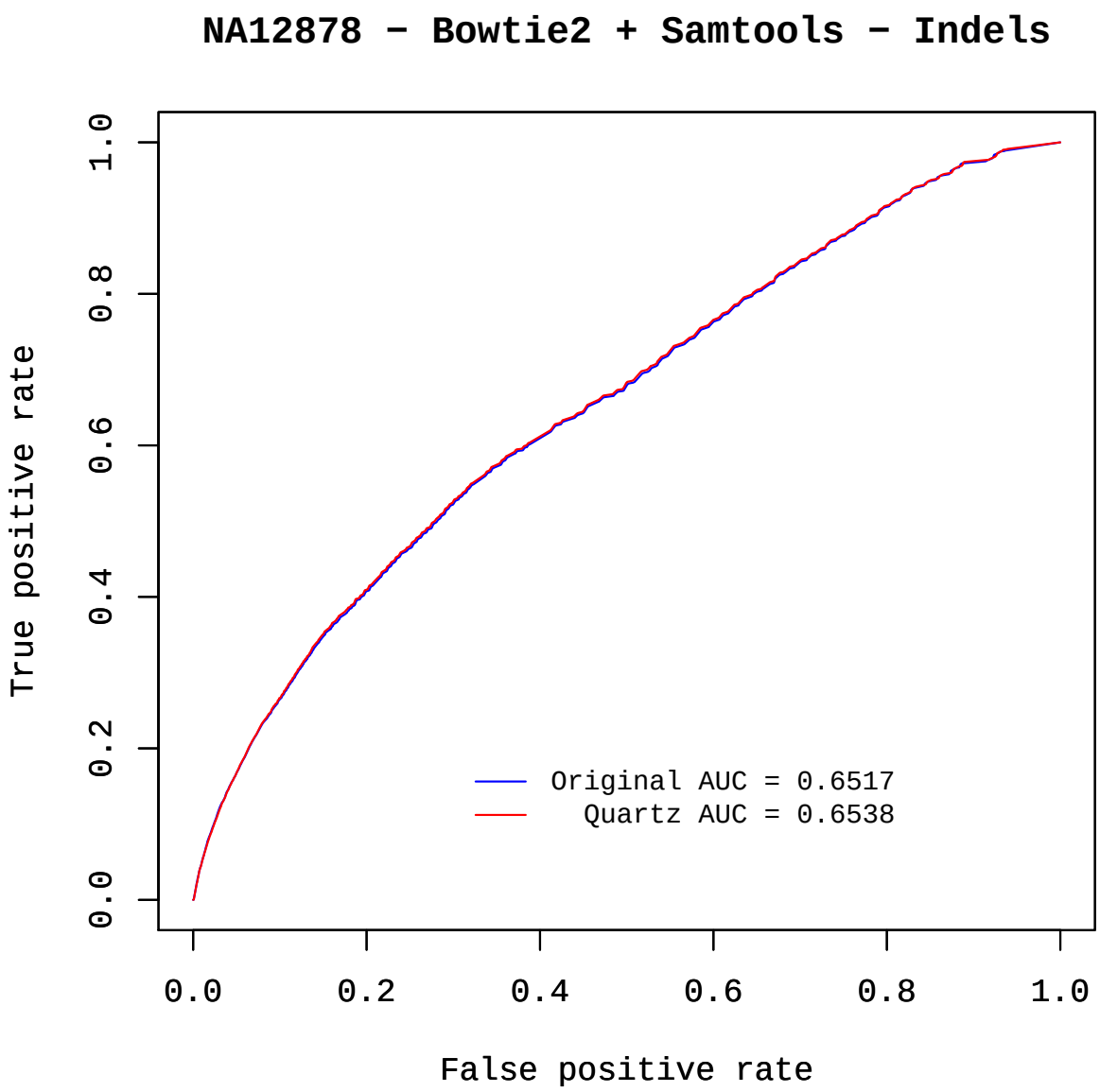


Figure S.10: Downstream indel calling before and after Quartz compression on the BWA + Samtools mpileup pipeline for NA12878.

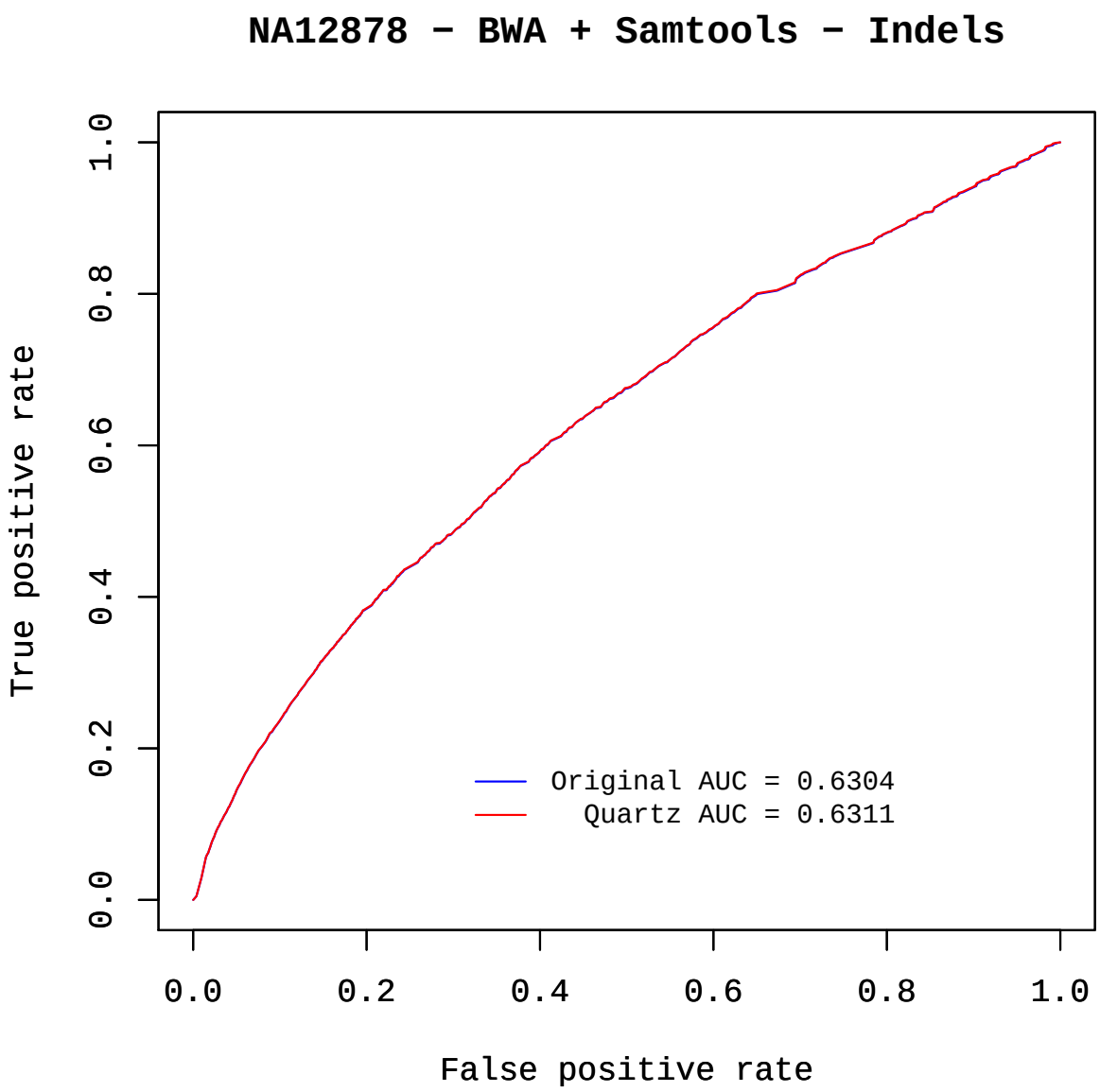
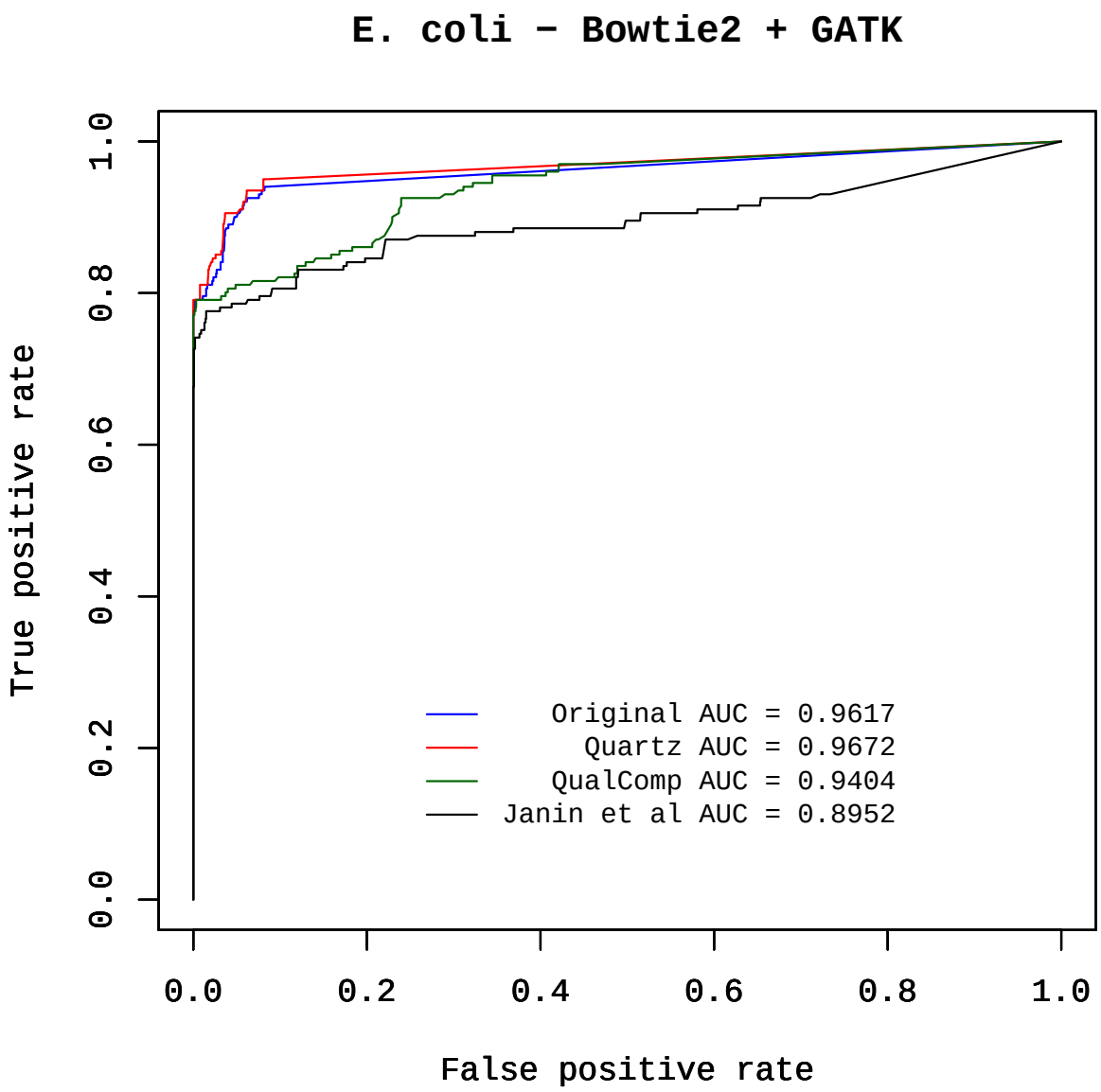


Figure S.11: Downstream indel calling before and after Quartz compression on the Bowtie 2 + Samtools mpileup pipeline for NA12878.

Quartz can be effectively applied to the *Escherichia coli* genome. All of the other results in this paper have been on the human genome, which is naturally of particular interest. However, while new k-mer dictionaries need to be generated for each species, Quartz can be fruitfully applied to compressing quality scores from other species, as demonstrated here in preliminary results with the K-12 MG1655 and DH10B strains of *E. coli*. For this experiment, we took paired-end sequencing reads of *E. coli* K-12 strain MG1655 using Illumina Genome Analyzer II from the European Nucleotide Archive [17] Study: ERP000092 and of *E. coli* K-12 strain DH10B using Illumina MiSeq, downloaded from the Orione database [18, 19]. The reference used was U00096.fna from the NIH Genbank for MG1655, and a truth set for DH10B as compared to MG1655 was generated by sampling one million 1000-mer sequences from the DH10B reference CP000948.fna and running Bowtie 2 and GATK Unified Genotyper on it. The dictionary was generated from the corpus of reads consisting of just the MG1655 and DH10B reads described above, such that all 32-mers with frequency at least 101 were included and only 32-mers with frequency at least 100 were included. Note that the depth-of-coverage for the corpus was ∼850x.

As Quartz performs best on low-coverage datasets, we subsampled the DH10B reads down to ∼9.5x depth-of-coverage for the analysis. In Figure S.12, we give SNP-calling accuracies for the uncompressed, Quartz-compressed, Qualcomp-compressed, and Janin et al-compressed quality scores, after attempting to match compression ratios. Note that this was possible for Qualcomp, which has an explicit bits/read parameter, but not for the method of Janin et al, so we chose a parameter set allowing the method of Janin et al additional bits to use.

As a caveat, the authors add that at higher depth-of-coverage, the improvement of Quartz over the uncompressed quality scores is reduced or sometimes even lost, in marked contrast to the human results, which always display improvement. This anomaly is likely due to the use of a low-quality dictionary; whereas the human corpus consisted of 97 different individuals, this corpus contains only 2 strains of *E. coli*. As less variation is encoded in the k-mer landscape of this corpus, Quartz thus has less information with which to improve variant calling accuracy. However, these preliminary results do suggest that Quartz is applicable to non-human genomes, though with the caveat that results will only be as good as the dictionary used.



Method	Bits/Q	Time (s)	AUC
Original / BZIP2	8 / 3.039	0 / 7	0.9617
Quartz	0.2587	7	0.9672
QualComp	0.2585	153	0.9404
Janin et al.	0.6477	383	0.8952

Figure S.12: Quartz compared to QualComp and the method of Janin et al. on a subsampling of reads from *E. coli* strain K12-DH10B to 9.5x depth-of-coverage. The method of Janin et al was run with parameters c=0, s=2, r=39, QualComp was run with bits per read set to 39.3, and Quartz was run with default replacement value Q50. For reference, performance after lossless compression by BZIP2 is also included with the results on the original scores.

References

[1] The 1000 Genomes Project Consortium. An integrated map of genetic variation from 1,092 human genomes. *Nature*, 491:1, 2012.

[2] Jayadev Misra and David Gries. Finding repeated elements. *Science of computer programming*, 2(2):143–152, 1982.

[3] Y William Yu, Deniz Yorukoglu, and Bonnie Berger. Traversing the k-mer landscape of NGS read datasets for quality score sparsification. In *Research in Computational Molecular Biology*, pages 385–399. Springer, 2014.

[4] Mark A DePristo, Eric Banks, Ryan Poplin, Kiran V Garimella, Jared R Maguire, Christopher Hartl, Anthony A Philippakis, Guillermo del Angel, Manuel A Rivas, Matt Hanna, et al. A framework for variation discovery and genotyping using next-generation DNA sequencing data. *Nature genetics*, 43(5):491–498, 2011.

[5] James K Bonfield and Matthew V Mahoney. Compression of FASTQ and SAM format sequencing data. *PloS one*, 8(3):e59190, 2013.

[6] Heng Li and Richard Durbin. Fast and accurate long-read alignment with Burrows–Wheeler transform. *Bioinformatics*, 26(5):589–595, 2010.

[7] Ben Langmead and Steven L Salzberg. Fast gapped-read alignment with Bowtie 2. *Nature methods*, 9(4):357–359, 2012.

[8] Heng Li, Bob Handsaker, Alec Wysoker, Tim Fennell, Jue Ruan, Nils Homer, Gabor Marth, Goncalo Abecasis, Richard Durbin, et al. The sequence alignment/map format and SAMtools. *Bioinformatics*, 25(16):2078–2079, 2009.

[9] Heng Li. Improving SNP discovery by base alignment quality. *Bioinformatics*, 27(8):1157–1158, 2011.

[10] Tobias Sing, Oliver Sander, Niko Beerenwinkel, and Thomas Lengauer. ROCR: visualizing classifier performance in R. *Bioinformatics*, 21(20):3940–3941, 2005.

[11] The 1000 Genomes Project Consortium. A map of human genome variation from population-scale sequencing. *Nature*, 467:1061–1073, 2010.

[12] Radoje Drmanac, Andrew B Sparks, Matthew J Callow, Aaron L Halpern, Norman L Burns, Bahram G Kermani, Paolo Carnevali, Igor Nazarenko, Geoffrey B Nilsen, George Yeung, et al. Human genome sequencing using unchained base reads on self-assembling DNA nanoarrays. *Science*, 327(5961):78–81, 2010.

[13] Aaron McKenna, Matthew Hanna, Eric Banks, Andrey Sivachenko, Kristian Cibulskis, Andrew Kernytsky, Kiran Garimella, David Altshuler, Stacey Gabriel, Mark Daly, et al. The Genome Analysis Toolkit: a MapReduce framework for analyzing next-generation DNA sequencing data. *Genome research*, 20(9):1297–1303, 2010.

[14] Geraldine A Auwera, Mauricio O Carneiro, Christopher Hartl, Ryan Poplin, Guillermo del Angel, Ami Levy-Moonshine, Tadeusz Jordan, Khalid Shakir, David Roazen, Joel Thibault, et al. From FastQ data to high-confidence variant calls: The Genome Analysis Toolkit best practices pipeline. *Current Protocols in Bioinformatics*, pages 11–10.

[15] Idoia Ochoa, Himanshu Asnani, Dinesh Bharadia, Mainak Chowdhury, Tsachy Weissman, and Golan Yona. QualComp: a new lossy compressor for quality scores based on rate distortion theory. *BMC Bioinformatics*, 14:187, 2013.

[16] Lilian Janin, Giovanna Rosone, and Anthony J Cox. Adaptive reference-free compression of sequence quality scores. *Bioinformatics*, 2013.

[17] Rasko Leinonen, Ruth Akhtar, Ewan Birney, Lawrence Bower, Ana Cerdeno-Tárraga, Ying Cheng, Iain Cleland, Nadeem Faruque, Neil Goodgame, Richard Gibson, et al. The european nucleotide archive. *Nucleic acids research*, page gkq967, 2010.

[18] Illumina Application Note. E. coli sequencing on the MiSeq system and ion torrent pgm system. 2012. [Online; accessed 07-Aug-2014].

[19] Gianmauro Cuccuru, Massimiliano Orsini, Andrea Pinna, Andrea Sbardellati, Nicola Soranzo, Antonella Travaglione, Paolo Uva, Gianluigi Zanetti, and Giorgio Fotia. Orione, a web-based framework for NGS analysis in microbiology. *Bioinformatics*, page btu135, 2014.